Prise en main Codes Apprentissage Supervisé

Ce document sert de guide **non-officiel** pour prendre en main les codes développés par Marielle Malfante disponible à l'adresse suivante : https://github.com/malfante/AAA . Si des points ne sont pas clairs n'hésitez pas à me contacter à l'adresse suivante : falcin@ipgp.fr, je me ferai un plaisir de vous aider.

Dans un premier temps, il est fortement conseillé de lire le fichier README.md.
Dans ce fichier vous trouverez :

- Les configurations et bibliothèques nécessaires pour faire tourner le code.
- Comment le prendre en main et l'exécuter (avec notamment les données d'exemple).
- Des explications sur les fichiers de configuration.
- Comment créditer les auteurs.
- Les références du papier pour lequel le code a été créé.

Ce documents s'organisera de la façon suivante :

1. Comment créer un environnement virtuel permettant de faire tourner les codes et quelles sont les bibliothèques requises.
2. L'application du code sur un jeu de donnée d'exemple et quelques explications sur les lignes de code
3. Les modifications à apporter pour faire fonctionner les codes avec de nouvelles données

Créer Environnement Virtuel

Suivre les instructions du fichiers README.MD :

This code was developed under Python 3, and needs the following libraries. Those libraries need to be previously installed.

- `numpy==1.13.3`
- `scipy==0.18.1`
- `pandas==0.19.2`
- `matplotlib==1.5.3`
- `numpy`
- `obspy==1.0.2`
- `python_speech_features==0.4`
- `sympy==1.0`
- `soundfile==0.8.1`

- `scikit-learn==0.18.1`

To install the correct version of python, along with the library, you can use miniconda environment manager.

1- Download and install miniconda: `https://conda.io/miniconda.html` NB: By installing miniconda, your `.bachrc` will be modified with the following line :

added by Anaconda3 4.3.0 installer

export PATH="/home/user/anaconda3/bin:$PATH"

We suggest you replace them by

"$PATH:/home/user/anaconda3/bin"

It will leave your computer configuration unchanged (in particular, your previous versions of python will still be used)

2- Create and activate your working environment (in a terminal session):

```
conda create -n AAA python=3.6

source activate myEnvName
```

3- Install the libraries:

```
pip install --upgrade pip

pip install -r AAA_requierements.txt.
```

4- Run the code (see next section)

5- Quit the working environment:

```
source deactivate
```

Jeu de Données d'Exemple

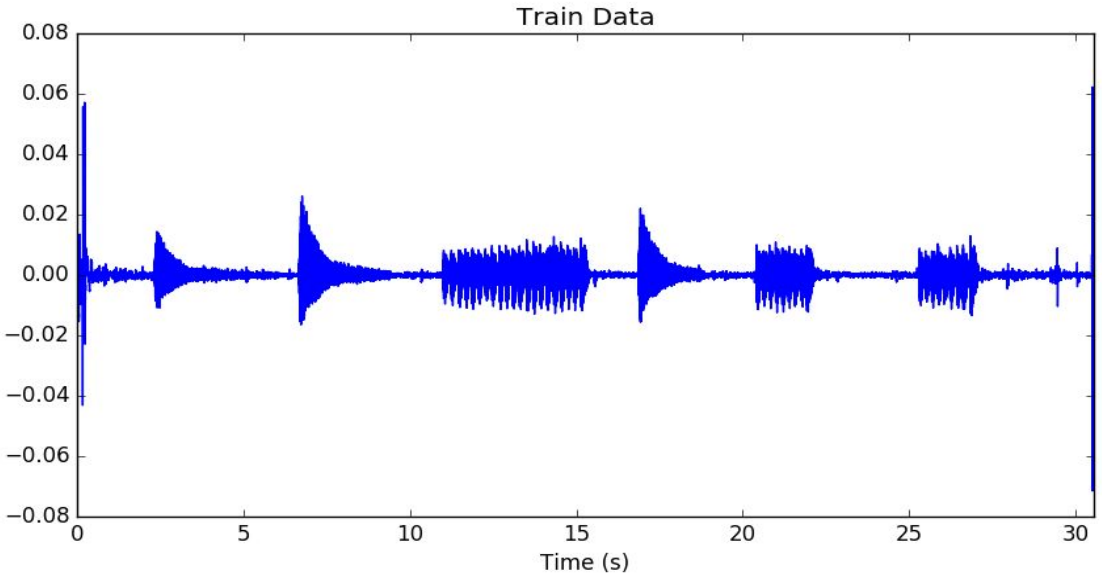
Dans un premier temps je vous invite à prendre en main les outils avec les données d'exemple fournies avec le reste des codes. Ce jeu de données, simple à comprendre et rapide à exécuter, permet de jouer avec les paramètres de configuration. Pour l'exécuter, vous avez juste à **modifier le project_root et écrire celui de votre machine dans le fichier de configuration générale : newsettings_10.json.**

Puis sous dans votre environnement virtuel exécutez la **commande :**
$ python3 PLAYGROUND1_CONTINUOUS_CLASSIFICATION.py

1. Données d'Exemple

Ce jeu de données se compose de 3 fichiers sonores en format .WAV.
Arborescence : */AAA-master/EXAMPLE/data/raw/example_*_data/**

Chaque fichier dure 30 secondes et on peut y écouter **3 classes de bruit : des accords, des arpèges et du bruit de fond**. Il y a un jeu de données d'entraînement labellisé (figure ci-dessous) et deux jeux de données de test non labellisés.



2. Catalogue Labellisé

Ici la labellisation de l'enregistrement consiste à découper le signal en fenêtres de 0.5 secondes et y attribuer une des 3 classes considérées. Le catalogue d'événements ici est un **catalogue regroupant la date, l'heure, la durée et la classe des événements ainsi que le chemin vers le fichier contenant l'événement**.

Ici notre catalogue brut est un fichier texte :
/AAA-master/EXAMPLE/data/shaping_data/catalogueBis.txt

Ce catalogue brut doit être formaté en **format pandas.dataframe pour être lisible par le soft**. Un notebook lisant le catalogue format texte et construisant un catalogue en format dataframe est disponible :
/AAA-master/EXAMPLE/data/shaping_data/00_catalogue_reading.ipynb

Le fichier ainsi créé est disponible :
/AAA-master/EXAMPLE/data/shaped/catalogueBis.pd

3. Fichiers de Configuration

Extrait du fichier README.md

All the settings related to a new project or a new run are indicated in a setting main setting.

It contains information regarding the project paths, the considered application, the signals preprocessing, the features used (linked to a dedicated feature configuration file), and the learning algorithms.

Extra information regarding the wanted analysis, the data to analyze and display parameters are indicated in a separate configuration file, which format depends on the usecase.

So, for each configuration, 3 configuration files are considered:

- the general setting file, contained in `config/general` folder
- the feature setting file, contained in `config/specific/features` folder
- the configuration file specific to the wanted analysis, contained in `config/specific/usecaseXX/`

Commented examples for each of the setting files are available (but keep in mind that json files do not support comments, so those files are simply there as examples.)

4. Exécution du Code

Je vous invite à exécuter la commande :
$ python3 PLAYGROUND1_CONTINUOUS_CLASSIFICATION.py
Je vais expliciter les différentes lignes de codes.
Pour l'instant les étapes c. et d. sont exécutables mais ne sont pas encore optimales.

a. Lecture du fichier de configuration

- *config = Config('../config/general/newsettings_10.json', verbatim=verbatim)*
- *config.readAndCheck()*

-> Lit les fichiers de configuration et vérifie l'existence des dossiers et fichiers requis pour l'exécution du script.

b. Création du modèle

- *analyzer = Analyzer(config, verbatim=verbatim)*
- *analyzer.learn(config)*

-> A partir des données labellisées ainsi que du choix d'algorithme classifieur (fichier de configuration général) on crée un modèle.
-> On performe une cross-validation (fichier de configuration général) en utilisant une partie des données labellisées pour l'entraînement et en utilisant le reste des données labellisées pour tester les performances du modèle. Si le paramètre verbatim > 0, le terminal affichera le score moyen et une matrice de confusion comparant les prédictions du modèle avec les véritables classes des événement du catalogue.

c. Application du modèle à un jeu de donnée non-labellisé

- *analyzedSet = Dataset(config,verbatim=verbatim)*
- *analyzedSet.analyze(analyzer, config, save=True)*
- *analyzedSet.makeDecision(config, save=True)*

-> Le modèle précédemment créé est appliqué à nouveau jeu de données en continu. Pour cela le signaux de test seront découpé en fenêtre suivant les paramètres enregistrés dans le fichier de configuration spécifique. Une prédiction sera faite pour chacune des fenêtres.

d. Affichage des prédiction faites sur le nouveau jeu de donnée

Pour afficher les prédictions faites sur le nouveau jeu de données, il y a deux méthodes, la première sans vérification des résultats et la deuxième avec vérification des résultats.

Méthode 1 :

- *analyzedSet.display(config, onlineDisplay=False, saveDisplay=True, forChecking=False)*

-> Affiche et enregistre les prédictions faites précédemment. Si la probabilité d'appartenance à une classe dépasse un certain seuil (fichier de config général) on attribue ladite classe sinon on attribue la classe inconnue. Figure disponible :
/AAA-master/EXAMPLE/res/10/fig/ Figure.png. Il y a une figure par enregistrement.

Méthode 2 :

- *analyzedSet.display(config, onlineDisplay=False, saveDisplay=True, forChecking=True, labelEncoder=analyzer.labelEncoder)*

-> Affiche et enregistre les prédictions faites précédemment. A l'inverse de la précédente ligne de code, celle-ci enregistre une figure par fenêtre et les enregistre dans des dossiers associés à la classe de la prédiction :
*/AAA-master/EXAMPLE/res/10/res_to_review/**to_review**/Classe/figure.png*
-> Il faudra alors vérifier si chacune des fenêtre a été correctement labellisée et après cela ranger la figure dans le fichier associés à sa vraie classe:
*/AAA-master/EXAMPLE/res/10/res_to_review/**reviewed**/Classe/figure.png*

- *trueLabels,predictedLabels=analyzedSet.getNumericalResults(config, analyzer.labelEncoder)*

-> Compare les labels prédit par le modèle avec les labels attribués manuellement à l'étape précédente.

Modifications pour un Nouveau Jeu de Données

Le code ainsi que les fichiers de configurations fournies ici sont réglés pour fonctionner avec le jeu de données d'exemple. Pour fonctionner sur un nouveau jeu de données des paramètres doivent être modifiés et des codes doivent être écrits.

1. Catalogues

Le catalogue brut dépend de chaque observatoire. Il faut extraire du catalogue la **date**, l'**heure**, la **durée** et la **classe** (affecté manuellement par l'observateur) de l'événement. Formater ensuite ces informations en format **pandas.dataframe**. Il faudra également savoir quel fichier sismique est associé à chacun des événements et renseigner dans le catalogue le **chemin** vers le fichier. Un exemple de script de formatage est disponible (*/AAA-master/EXAMPLE/data/shaping_data/00_catalogue_reading.ipynb)*
mais celui-ci ne marche que pour le fichier catalogueBis.txt (le catalogue brut du jeu d'exemple).

2. Fonction de Lecture

Pour chaque jeu de données il faudra écrire une fonction de lecture dans le script :
automatic_processing/DataReadingFunction.py
La fonction que vous écrirez prendra **en argument le chemin vers le fichier** enregistré dans le catalogue et **rendra en sortie la forme d'onde, la fréquence d'échantillonnage, le temps du début et de fin du fichier et le nombre d'échantillons.** Cette fonction fonctionnera de paire avec la fonction *requestObservation* dans le même script qui s'occupera d'extraire seulement l'événement d'intérêt.

Il se peut que selon le format de vos données cette façon de récupérer l'événement ne soit pas optimale (ce fut le cas pour moi avec les fichiers .mseed subdivisés en stream), il faudra alors créer une nouvelle fonction combinant *requestObservation* et celle que je vous ai précédemment invité à écrire et à remplacer dans *analyzer.py* à la ligne 130 la fonction *requestObservation* pour votre nouvelle fonction. Cette nouvelle fonction devra **extraire du fichier directement la forme d'onde liée à l'événement.**

3. Fichiers de configuration

Dans le fichier de **configuration général**, il vous faudra modifier le **chemin vers le catalogue** et le **nom de l'application**.

Dans le fichier de **configuration spécifique**, il faudra modifier le **chemin vers les données à tester**, le **type de données**, la **fonction de lecture**, les **paramètres de l'analyse en continue** et les **paramètres du spectrogramme**.